FTI-SLAM: Federated Learning Enhanced Thermal-Inertial SLAM

Haochen Liu[*]
Department of Computer Science
University of Cambridge
hl663@cam.ac.uk

Hantao Zhong[†]
Department of Computer Science
University of Cambridge
hz445@cam.ac.uk

Abstract

Utilising thermal images for Simultaneous Localization and Mapping has effectively improved the system's performance in tracking a robot's movement and detecting loop closures. However, for centralised training, transmitting a large amount of visual data not only requires high bandwidth and storing devices but also raises privacy concerns, as images might contain sensitive information. For generalisation, it is essential to expand the input domain to more realistic scenarios rather than relying on experimental environments. To deal with these concerns, we implemented FTI-SLAM by applying federated learning to Thermal–Inertial SLAM. We conducted a series of experiments and showed that federated learning is feasible for the task and can improve overall performance.

1 Introduction

Simultaneous Localization and Mapping (SLAM) is an essential field in robotics. For robots and other autonomous systems, it enables them to confirm their locations and moving trajectories without relying on prior knowledge of the environment or the support of positioning systems [1]. SLAM is a sophisticated task due to the uncertainties of the environment and sensors, and and high computational complexity of existing algorithms [2, 3]. It has broad application scenarios, including domestic robots, autonomous vehicles, and support in emergency and hazardous missions.

The setup of a SLAM system involves a front end that uses sensors to collect, process, and transform data, and a back end that utilises output provided by the front end for the estimation of states. Early SLAM approaches relies heavily on filters such as particle filter and extended Kalman filter, and sensors that only provided a limited amount of information [4]. In recent decades, researchers focused on visual and 3D structural sensors such as light detection and ranging (LiDAR), and remarkable progress has been made with the rapid evolution of deep learning [5, 6]. Systems like oriented FAST and rotated BRIEF (ORB)-SLAM and LOAM are successful examples of SLAM that adopted RGB and LiDAR sensors, showing the capability of operating in both indoor and outdoor environments with robustness [7].

Though widely used, visual-based SLAM has some obvious weaknesses. One of them is the high sensitivity to different illumination conditions. In dark environments, RGB sensors cannot provide data with acceptable quality, which affects the performance of the back end [1]. Though LiDAR sensors are less prone to change of illumination, weather conditions such as rain and fog will drastically reduce the quality of LiDAR sensing [8]. Similarly, in complex environments, for example, in a scene of fire with lots of smoke, the moving particles will downgrade the LiDAR sensors' ability to recognise the environment.

[*] Wolfson College, University of Cambridge, CB3 9BB

[†] Clare Hall College, University of Cambridge, CB3 9AL

Based on those findings, researchers tend to seek more robust sensing approaches, and one of the directions is building a thermal-inertial SLAM system. Thermal cameras are less prone to the common weakness of RGB and LiDAR sensing, as they focus on the long-wave infrared portion of the spectrum, therefore it is commonly used in low-visibility scenarios. Saputra et al. applied it with deep neural networks (DNN) and designed DeepTIO, the first deep thermal-inertial odometry involving visual hallucination, providing accurate odometry estimations [9]. This work was later expanded to SLAM by Saputra et al. to SLAM, proposing the first graph-based thermal-inertial SLAM (TI-SLAM) system [10].

Systems like TI-SLAM require a large amount of time and resources for training. In practical user scenarios, the domain of data might vary from case to case, which requires a generalised model for the task. In addition, due to the sensitivity concerns and the difficulty of transmitting a large amount of data with a relatively big size, training on users' ends should be considered. Inspired by these, we implemented and trained the federated learning version of TI-SLAM. We conducted experiments in different setups and compared the performance with the original implementation. Our contributions can be summarised as follows.

1) We adopted the materials from the module, provided a complete implementation of federated learning adapted TI-SLAM, and conducted a series of experiments on it under the limitation of computational resources and difficulties of adapting the source code and environments.

2) We presented the experimental results with detailed analysis and pointed out the possible direction of improvement for future works.

2 Background

2.1 SLAM

Previously, SLAM has evolved from algorithm-based to DNN-oriented. Early works such as MonoSLAM rely on algorithms including extended Kalman filter to estimate position and poses frame by frame[11]. Keyframe-based bundle adjustment approach, including parallel tracking and mapping (PTAM) and ORB-SLAM, became more widely used later due to higher accuracy [12, 7]. These methods do perform well in some cases, but their sensitivity to outliers and dependency on hand-engineered features restricted their ability of generalisation [13, 14].

During the last few years, learning-based approaches have been replacing algorithm-based ones, and DNN has become popular in SLAM front ends. The reason is the extra robustness brought by learning from large-scale data rather than relying on hand-engineered features [10]. The DNN approaches focused on different aspects of tasks such as odometry, relocalisation, and loop closure detection.

Odometry is about keeping track of positions throughout the time [15]. DeepVO, the first attempt of odometry with DNN, adopts deep recurrent convolutional neural networks (RCNN) to predict positions based on RGB images [16]. More DNN approaches emerged after the success of DeepVO, but none of them involved thermal images before DeepTIO [9].

Place recognition, also known as loop closure detection (LCD) in SLAM, is the task responsible for checking whether the robot is located at a place visited before [17]. Many visual place recognition methods, such as DeCAF, adopted convolutional neural networks (CNN) for feature extraction, and these models outperform models relying on handcrafted features[18]. Existing thermal-based place recognition such as fast appearance-based mapping are still based on standard approaches with the assistance of other dimensions such as RGB image data [19].

Relocalisation refers to constraining loop closure, and it is essential for dealing with lost or kidnapped robot problem - a situation in which the robot is carried to random locations [20]. PoseNet is a work in this area that is capable of providing real-time response with error within 2 meters and 3 degrees, leveraging transfer learning and CNN to complete the deep CNN camera pose regressor[21].

By combining the existing works in these aspects, a complete SLAM system will be constructed. DeepSLAM is one of those works. It is a combination of odometry, mapping, and place recognition, and it achieved good performance in public visual odometry benchmark [22]. Though other works focusing on combining different approaches exist, few works regarding applying thermal images like TI-SLAM exist.

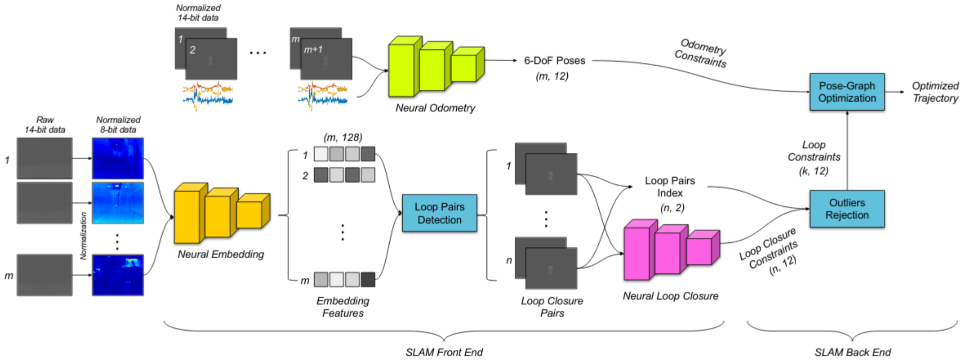


Figure 1: TI-SLAM high-level architecture

2.2 DeepTIO

DeepTIO is the first odometry tool that utilises thermal cameras to deal with low-visibility environments [9]. It encodes the thermal data and hallucinates visual features with a CNN, and uses a recurrent neural network (RNN) to capture the temporal dependencies within inertial measurement unit data. TI-SLAM was built on DeepTIO [10].

2.3 Federated Learning and SLAM

The concept of federated learning is about local computing and model transmission, ensuring the original data is processed and utilised on clients' side in order to mitigate the concern of privacy [23]. This is especially important for SLAM, as it relies heavily on RGB images and LiDAR data that might contain sensitive information about the environment and users. FC-SLAM is one of the few federated learning solutions for SLAM, enhancing the quality and robustness of feature extraction effectively utilising the computational resources available in the cloud server to cooperatively improve the overall performance[24]. There has not been any work focusing on federated learning, thermal images, and SLAM at the same time.

3 Methodology

3.1 Baseline Model and Problem Formulation

To enhance our focus on the application of federated learning within a SLAM system, we have adapted the TI-SLAM framework [10], a thermal-inertial SLAM system as our baseline model. TI-SLAM employs thermal imaging combined with a graph-based approach and a probabilistic neural network, significantly enhancing the performance of visual SLAM systems in low-light or visually obscured environments, such as smoke. The TI-SLAM model comprises three primary components: the Neural Odometry Network, the Neural Embedding Network, and the Neural Loop Closure Network. The Neural Thermal-Inertial Odometry Network is an odometry module that processes a series of thermal images to compute the 6-DoF poses over time. The Neural Embedding Network generates a global embedding descriptor, subsequently utilized for Loop Pairs Detection by evaluating the discrepancy among embedding using cosine distance. The Neural Loop Closure Network calculates loop closure constraints and ascertains whether the mobile agent has revisited a specific location. The final trajectory is optimized using these loop closure constraints and 6-DoF poses through the Levenberg-Marquardt algorithm [25]. A summary of the TI-SLAM structure is presented in Figure1.

Given that TI-SLAM is divided into three subnetworks, the training process is similarly divided into three independent network training stages. The Neural Odometry Network, Neural Embedding Network, and Neural Loop Closure Network are each trained independently. This divergence from a classical single network approach in federated learning presents a unique challenge in adapting a federated learning framework and developing a global model updating strategy. For this project, we will focus on exploring potential federated learning strategies across the three subnetworks, with varying dataset domains. We have restructured the TI-SLAM training process into three independent federated learning challenges, where the clients in each subnetwork share the same dataset across all three federated learning training scenarios. Our objective is to investigate, within a federated learning context, which strategies are most effective for each subnetwork to yield the optimal SLAM model. This is particularly challenging given that each client possesses only a subset of the dataset. Such an approach could significantly benefit real-world applications, such as autonomous driving, where most state-of-the-art SLAM models are not end-to-end, and agents typically operate in diverse environments. Our federated learning-adapted version of TI-SLAM is referred to as FTI-SLAM for simplicity in later discussions.

3.2 Implementation of Federated Learning and training

In our study of federated learning, we have used the Flower framework [26] for implementing the federated learning component of FTI-SLAM. Flower is a federated learning framework that easy to simulate various federated learning setups. Although Flower supports TensorFlow 2.0 [27], the original TI-SLAM codebase was based on TensorFlow 1.0. Consequently, we have reimplemented and adapted the TI-SLAM codebase to TensorFlow 2.6 and Keras 2.6 [28], with the correctness of our implementation to be discussed subsequently. The implementation of FTI-SLAM is available as a GitHub repository[3].

As mentioned above, the training process can be divided into three subnetwork training problems. We established three independent federated learning systems for each subnetwork. Each client is configured to have an identical dataset across all three subnetworks. This approach is closely reflecting real-world scenarios, where different clients might operate in varying indoor environments. The implementation was carried out using the Flower framework to simulate the federated learning workload.

The Neural Embedding Network training in FTI-SLAM follows the same process as in TI-SLAM, training from scratch with anchor, positive, and negative images for embedding training.

The training process for the Neural Odometry Network and Neural Loop Closure Network differs from TI-SLAM. For the Neural Odometry Network, it is necessary first to initialize a hallucination network using Flownet [29], followed by hallucination training, before continuing to the thermal network training. Unfortunately, we were unable to locate a Flownet pretrained weight compatible with TI-SLAM hallucination network and the authors of TI-SLAM didn't provide any details on that. To address and simplify this issue, we omitted the hallucination network training and initialized it using the best checkpoint provided by the TI-SLAM authors. In FTI-SLAM, the Neural Loop Closure Network begins training with a pretrained hallucination weight loaded into the global model, which is from TI-SLAM best checkpoint as well.

For the Neural Loop Closure Network in TI-SLAM, the trained Neural Odometry Network is used to initialize the feature extractor. This design shares the same feature extractor between the Neural Odometry Network and the Neural Loop Closure Network. However, rewriting the training processes of both networks in FTI-SLAM with a federated learning update strategy for simultaneous feature extractor updating would be overly complex. Similar to our approach with the Neural Odometry Network, we utilized the best checkpoint from the TI-SLAM codebase to initialize the feature extractor for the global model at the outset.

Due to constraints in computational resources, we reduced the precision of image data from float64 to float16. This reduction conserves VRAM during training and inference phases. A summary of the FTI-SLAM structure is illustrated in Figure 2.

4 Experiment

4.1 Dataset and setup

The dataset used in this study was collected by TI-SLAM authors with a ground moving robot (Turtlebot 2). This robot is equipped with a thermal camera and an IMU sensor, and the data were collected in an indoor environment. Due to computational resources limitations, we selected 6

[3] https://github.com/MrTooOldDriver/ti-slam/

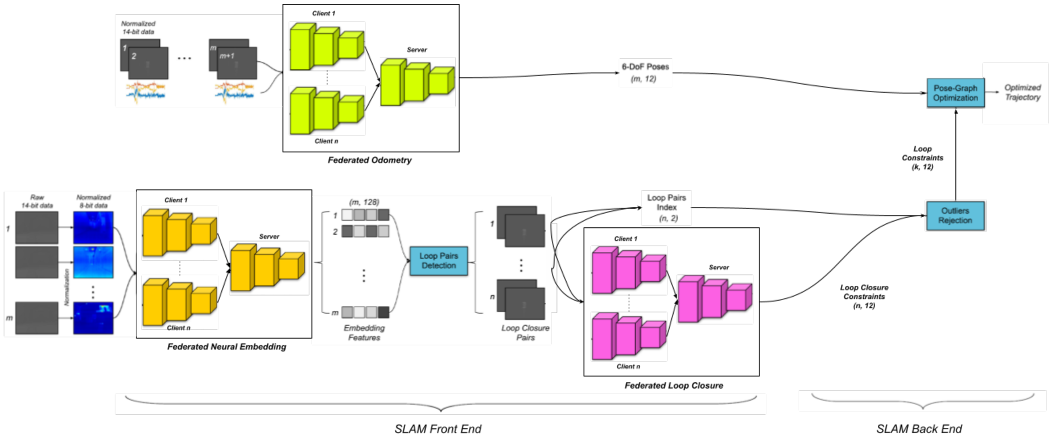


Figure 2: FTI-SLAM structure

sequences for training and 3 sequences for testing, allowing us to keep the training duration within a reasonable timeframe. Given the relatively smaller size of the dataset, we reduce the training epochs to just 50.

All training and inference tasks for FTI-SLAM were performed using NVIDIA P4 and NVIDIA RTX 2080Ti graphics cards. The learning rate and batch size settings were the same as TI-SLAM configuration. Except where specifically stated, all experimental settings were identical to TI-SLAM paper.

In the results section, RMSE refers to Root Mean Square Error, and ATE stands for Absolute Trajectory Error. The methodologies for calculating these metrics are consistent with TI-SLAM paper.

	Odom		Embedding		Loop	
Setup	Clients	val_loss	Clients	val_loss	Clients	val_loss
Original - w data selection - 0.045	1	-35.15777	1	0.70846	1	-6.34729
Tf2 - w data selection - 0.045	1	-33.76708	1	0.72218	1	-3.02713
Tf2 - w data selection	1	-33.76708	1	0.72218	1	-3.02713
Tf2 - odom fl	2	-32.72081	1	0.72218	1	-3.02713
Tf2 - loop fl	1	-33.76708	1	0.72218	2	-3.43019
Tf2 - embedding fl	1	-33.76708	2	0.43718	1	-3.02713
Tf2 - all fl	2	-32.72081	2	0.43718	2	-3.43019
Tf2 - loop fl - 5round10epochs	1	-33.76708	1	0.72218	2	-9.36247
Tf2 - loop fl - 5round10epochs all fl	2	-32.72081	2	0.43718	2	-9.36247

Table 1: Training details of subnetworks

4.2 TI-SLAM vs TI-SLAM tf2

To validate our implementation of TI-SLAM in TensorFlow 2, we first trained both the original TI-SLAM and our TensorFlow 2 version using the same dataset and settings. The results are presented in Tables 1, 2, 3, and 4. In these tables, the term *Original - w data selection - 0.045* refers to the official implementation of TI-SLAM, while *Tf2 - w data selection - 0.045* denotes our TensorFlow 2 implementation of TI-SLAM. Both models were trained with the dataset we selected. The "0.045" refer to the loop threshold which is the same as TI-SLAM.

From the results, it is clear that except for the embedding training, both the odometry and loop closure components in the TensorFlow 2 implementation show a decline in training performance compared to the original implementation. We believe this is mainly due to our decision to reduce the precision of the input thermal images from float64 to float16 as a result of computational resource limits.

	Seq 27			
Setup	RMSE ATE (m)	Variance ATE (m)	RMSE (m)	Improvement(%)
Original - w data selection - 0.045	0.9898	0.0928	1.4178	30.1840
Tf2 - w data selection - 0.045	0.9163	0.2716	0.6984	-31.1931
Tf2 - w data selection	**0.6218**	0.2336	**0.6985**	10.9798
Tf2 - odom fl	0.7053	0.3836	0.7484	5.7506
Tf2 - loop fl	0.5944	0.2266	**0.6985**	**14.9084**
Tf2 - embedding fl	0.6276	**0.0919**	**0.6985**	10.1482
Tf2 - all fl	0.6664	0.0981	0.7484	10.9492
Tf2 - loop fl - 5round10epochs	0.6373	0.2510	**0.6985**	8.7666
Tf2 - loop fl - 5round10epochs all fl	0.7093	0.1140	0.7484	5.2139

Table 2: Evaluation Result For Sequence 27. Best result apart from original are shown in **bold**

	Seq 28			
Setup	RMSE ATE (m)	Variance ATE (m)	RMSE (m)	Improvement(%)
Original - w data selection - 0.045	1.8408	0.6385	1.0240	-79.7696
Tf2 - w data selection - 0.045	0.8174	0.1349	0.6128	-33.38
Tf2 - w data selection	1.0118	0.2229	**0.6129**	-65.0881
Tf2 - odom fl	1.0609	0.2613	0.7790	**-36.1961**
Tf2 - loop fl	0.9264	0.1900	**0.6129**	-51.1561
Tf2 - embedding fl	**0.9063**	**0.1516**	**0.6129**	-47.8681
Tf2 - all fl	1.5409	0.4941	0.7790	-97.8164
Tf2 - loop fl - 5round10epochs	1.0963	0.4964	**0.6129**	-78.8763
Tf2 - loop fl - 5round10epochs all fl	1.5367	0.5186	0.7790	-97.2774

Table 3: Evaluation Result For Sequence 28. Best result apart from original are shown in **bold**

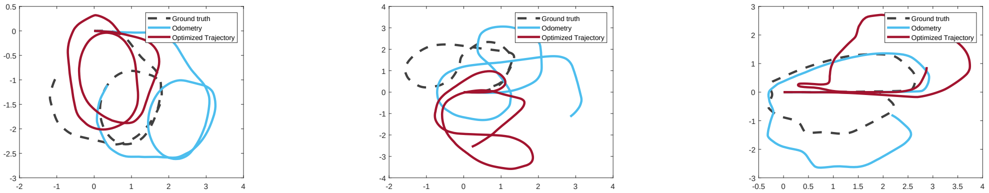
	Seq 29			
Setup	RMSE ATE (m)	Variance ATE (m)	RMSE (m)	Improvement(%)
Original - w data selection -	0.7012	0.0958	0.8035	12.7286
Tf2 - w data selection - 0.045	1.1573	0.4674	0.4115	-1.811
Tf2 - w data selection	**0.5818**	0.1186	0.4115	**-41.3484**
Tf2 - odom fl	0.5819	**0.1159**	**0.3699**	-57.3170
Tf2 - loop fl	0.6307	0.6307	0.4116	-53.2329
Tf2 - embedding fl	1.0695	0.4940	0.4116	-159.8473
Tf2 - all fl	0.9586	0.3608	**0.3699**	-159.1609
Tf2 - loop fl - 5round10epochs	0.7200	0.1234	0.4116	-74.9202
Tf2 - loop fl - 5round10epochs all fl	1.0292	0.4184	**0.3699**	-178.2316

Table 4: Evaluation Result For Sequence 29. Best result apart from original are shown in **bold**

Despite this reduction, the TensorFlow 2 version appears to produce meaningful results. However, the evaluation results indicate a noticeable decrease in SLAM performance improvements in the TensorFlow 2 version, which is likely again, due to the reduced precision of the thermal image inputs.

4.3 FTI-SLAM subnetwork FL training

In our federated learning experiments, we carried out with various combinations of traditional non-federated and federated learning training across subnetworks to observe potential overall SLAM improvements. The results are detailed in Tables 1, 2, 3, and 4. In all federated learning experiments, we reduced the embedding threshold to 0.005 to enforce constraints on loop detection over embedding



(a) seq27 (b) seq28 (c) seq29

Figure 3: Optimized Trajectory Result of *Tf2 - all fl*

cosine distances. This adjustment was necessary to minimize false positive loops, which lead significantly impact pose graph optimization time. For these federated learning experiments, we utilized a configuration of 2 clients, 50 rounds, and 1 epoch per round, while each client having access to only half of the training and validation datasets. The global model was updated using a federated weight averaging strategy, where the weights were calculated based on loss. The setup names refer to which subnetworks were trained using federated learning. *Tf2 - all fl* denotes results from all three subnetworks trained with federated learning. *Tf2 - w data selection* refers to standard TensorFlow 2 TI-SLAM training.

The results demonstrate that federated learning significantly improves the embedding training process, reducing the validation loss to 0.43718. However, the federated learning approach in the odometry network did not present any significant improvements. Interestingly, the loop closure detection network under federated learning exhibited poorer training performance, suggesting that the embedding network benefits most from this approach.

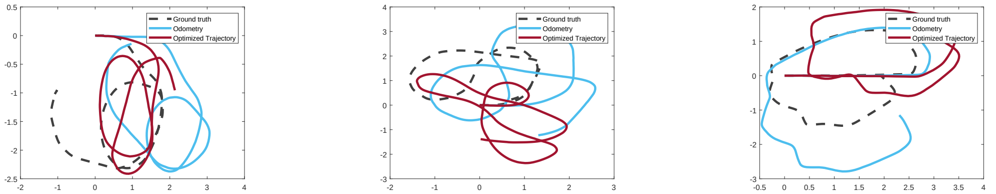
Moreover, SLAM evaluation across all validation sequences tells a different story, with no clear pattern of federated learning improvements observed. Across most sequences, applying federated learning to the original setup usually led to improvements in RMSE and ATE. These improvements varied depending on the sequence and specific setup configuration. The most notable results, apart from the original, were often achieved by modifying the loop or embedding layers, indicating these areas as potential beneficiaries of federated learning in this context. However, it is important to note that some federated learning setups resulted in significant performance degradation, particularly in Sequences 28 and 29, suggesting that certain configurations may be counterproductive.

In conclusion, although there is no strong evidence of federated learning enhancing overall SLAM performance, it does appear to improve some subnetwork training aspects. The lack of clear performance improvement in SLAM evaluations and the link with the mismatch between validation loss and SLAM evaluation results could be attributed to the small range dataset. This limitation was primarily due to computational resource constraints, as a complete federated learning simulation took up to 6 hours per subnetwork training. Therefore, it is recommended to extend the training with more datasets. Moreover, the embedding network seems to benefit the most from federated learning, likely because it was the only model trained from scratch without any weight initialization. For finetuning training style networks, such as the Neural Odometry Network and Neural Loop Closure training, federated learning contributed minimally or not at all to the final model. In some cases (e.g., Sequence 28), federated learning improved performance by up to 29%.

4.4 FTI-SLAM Federated Learning Strategy

To further investigate federated learning training strategies, we conducted an additional experiment with different settings for rounds and epochs per round. Specifically, we trained the loop closure network with 5 rounds, each round has 10 epochs. The results are presented in Tables 1, 2, 3, and 4, and are denoted as *Tf2 - loop fl - 5round10epochs*.

The outcomes from this configuration showed a significant improvement in the loop closure training process, with the validation loss decreasing to -9.39. However, this improvement in training did not translate into improved results in SLAM evaluation. This again outlined the vast gap between SLAM evaluation metrics and validation loss. It suggests that a more extensive and diverse dataset might be necessary to more accurately reflect the overall data distribution in future research. Moreover,



(a) seq27

(b) seq28

(c) seq29

Figure 4: Optimized Trajectory Result of *Tf2 - loop fl - 5round10epochs all fl*

this observation indicates that while specific federated learning strategies could optimize certain training metrics, their effectiveness in enhancing real-world SLAM performance remains uncertain. Therefore, future studies should aim to align federated learning optimizations more closely with practical SLAM improvements.

5 Conclusion

In conclusion, although this paper did not identify a novel federated learning approach for a non-end-to-end trainable SLAM model, it does highlight potential areas for improvement, such as neural embedding, which may benefit from federated learning techniques. For future work, enhanced computational resources are essential to the broader exploration of various federated learning update strategies and to incorporate more datasets in the training process. This study was significantly constrained by computational limitations, as complete training of all subnetworks with federated learning required over 18 hours. Future research should also focus on the shared feature extractor between the Neural Odometry Network and the Neural Loop Closure Network. Optimizing the feature extractors of both networks jointly through federated learning could bring substantial improvements. Additionally, further investigation is required to understand the impact of federated learning on the hallucination network and its subsequent effects on the second stage of training, which could be critical. Overall, this paper suggests some directions for federated learning in SLAM models and outlines the need for more robust computational resources to fully explore these potentials.

References

[1] Hamid Taheri and Zhao Chun Xia. “SLAM; definition and evolution”. In: *Engineering Applications of Artificial Intelligence* 97 (2021), p. 104032. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2020.104032.

[2] Farshid PirahanSiah, Siti Abdullah, and Shahnorbanun Sahran. “Simultaneous Localization and Mapping Trends and Humanoid Robot Linkages”. In: *Asia-Pacific Journal of Information Technology and Multimedia* 2 (Dec. 2013), p. 12. DOI: 10.17576/apjitm-2013-0202-03.

[3] Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. “On the complexity and consistency of UKF-based SLAM”. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 4401–4408. DOI: 10.1109/ROBOT.2009.5152793.

[4] F. Lu and E. Milios. In: *Autonomous Robots* 4.4 (1997), pp. 333–349. ISSN: 0929-5593. DOI: 10.1023/a:1008854305733.

[5] Iman Abaspur Kazerouni et al. “A survey of state-of-the-art on visual SLAM”. In: *Expert Systems with Applications* 205 (2022), p. 117734. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.117734.

[6] Misha Urooj Khan et al. “A Comparative Survey of LiDAR-SLAM and LiDAR based Sensor Technologies”. In: *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*. 2021, pp. 1–8. DOI: 10.1109/MAJICC53071.2021.9526266.

[7] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[8] M. Kutila et al. “Automotive LiDAR performance verification in fog and rain”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 1695–1701. DOI: 10.1109/ITSC.2018.8569624.

[9] Muhamad Risqi Utama Saputra et al. “DeepTIO: A Deep Thermal-Inertial Odometry with Visual Hallucination”. In: *CoRR* abs/1909.07231 (2019). arXiv: 1909.07231. URL: http://arxiv.org/abs/1909.07231.

[10] Muhamad Risqi Utama Saputra et al. “Graph-based Thermal-Inertial SLAM with Probabilistic Neural Networks”. In: *CoRR* abs/2104.07196 (2021). arXiv: 2104.07196. URL: https://arxiv.org/abs/2104.07196.

[11] Andrew J. Davison et al. “MonoSLAM: Real-Time Single Camera SLAM”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1052–1067. DOI: 10.1109/TPAMI.2007.1049.

[12] Georg Klein and David Murray. “Parallel Tracking and Mapping for Small AR Workspaces”. In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852.

[13] Muhamad Risqi U. Saputra, Andrew Markham, and Niki Trigoni. “Visual SLAM and Structure from Motion in Dynamic Environments: A Survey”. In: *ACM Comput. Surv.* 51.2 (Feb. 2018). ISSN: 0360-0300. DOI: 10.1145/3177853.

[14] Sen Wang et al. “End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks”. In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 513–542. DOI: 10.1177/0278364917734298. eprint: https://doi.org/10.1177/0278364917734298.

[15] Mohammad O. A. Aqel et al. “Review of visual odometry: types, approaches, challenges, and applications”. In: *SpringerPlus* 5.1 (Oct. 2016). ISSN: 2193-1801. DOI: 10.1186/s40064-016-3573-7.

[16] Sen Wang et al. “Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks”. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 2043–2050.

[17] Xiwu Zhang, Lei Wang, and Yan Su. “Visual place recognition: A survey from deep learning perspective”. In: *Pattern Recognition* 113 (2021), p. 107760. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2020.107760.

[18] Jeff Donahue et al. “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”. In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 647–655. URL: https://proceedings.mlr.press/v32/donahue14.html.

[19] Mark Cummins and Paul Newman. “FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance”. In: *The International Journal of Robotics Research* 27.6 (2008), pp. 647–665. DOI: 10.1177/0278364908090961.

[20] S.P. Engelson and D.V. McDermott. “Error correction in mobile robot map learning”. In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. 1992, 2555–2560 vol.3. DOI: 10.1109/ROBOT.1992.220057.

[21] A. Kendall, M. Grimes, and R. Cipolla. “PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2015, pp. 2938–2946. DOI: 10.1109/ICCV.2015.336.

[22] Ruihao Li, Sen Wang, and Dongbing Gu. “DeepSLAM: A Robust Monocular SLAM System With Unsupervised Deep Learning”. In: *IEEE Transactions on Industrial Electronics* 68.4 (2021), pp. 3577–3587. DOI: 10.1109/TIE.2020.2982096.

[23] Chen Zhang et al. “A survey on federated learning”. In: *Knowledge-Based Systems* 216 (2021), p. 106775. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2021.106775.

[24] Zhaoran Li et al. “FC-SLAM: Federated Learning Enhanced Distributed Visual-LiDAR SLAM In Cloud Robotic System”. In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2019, pp. 1995–2000. DOI: 10.1109/ROBIO49542.2019.8961798.

[25] Ananth Ranganathan. “The levenberg-marquardt algorithm”. In: *Tutoral on LM algorithm* 11.1 (2004), pp. 101–110.

[26] Daniel J Beutel et al. “Flower: A friendly federated learning research framework”. In: *arXiv preprint arXiv:2007.14390* (2020).

[27] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[28] François Chollet et al. *Keras*. https://keras.io. 2015.

[29] Philipp Fischer et al. “Flownet: Learning optical flow with convolutional networks”. In: *arXiv preprint arXiv:1504.06852* (2015).